Strategies for Enabling Software Reuse within the Earth Science Community

** * This paper will be presented at IGARSS '04 and will be published in the conference proceedings ***

Samadi, Shahin (Shahin.Samadi@gsfc.nasa.gov); Alameh, Nadine (nadine.alameh@gst.com); Wolfe, Robert (robert.wolfe@gsfc.nasa.gov); Olding, Steve (solding@everware.com); Isaac, David (david.isaac@teambps.com)
NASA Goddard Space Flight Center
Greenbelt, MD

Abstract— **The Earth Sciences software development community is often challenged to provide cost effective, highly reliable and easy-to-use software to achieve scientific missions. In the process, the NASA Earth Science Enterprise (ESE) spends a significant amount of resources developing software components and other software development artifacts that may also be of value if reused in other projects requiring similar functionality. A recent study performed under the NASA's Strategic Evolution of ESE Data Systems (SEEDS) initiative suggests that reuse of ESE software can drive down the cost and time of system development, increase flexibility and responsiveness of these systems to new technologies and requirements; and increase effective and accountable community participation. In 2004, the Earth Science Software Reuse Working Group was created to oversee the development of a process that will maximize the reuse potential of existing software components while recommending strategies for maximizing the reusability potential of yet-to-be-designed components.**

Software reuse; reusability; classification; Earth Science; SEEDS; NASA.

I. Introduction

The Earth Science community has invested heavily in developing many software systems that range from simple scientific algorithms to large and complex information systems that store, process, analyze, and disseminate vast volumes of remote sensing information. These systems represent a tremendous potential source of software that could be reused to create new systems and enhance current ones to meet future mission needs within given budget constraints. Software reuse has many obvious benefits such as increased productivity, reduced time to market and improved quality. Yet, realizing these benefits for Earth science data systems has been challenging. Although new generations of the more complex systems often exploit domain knowledge and expertise from previous development activities, a more disciplined reuse approach is still needed to further assist with cost reduction and productivity improvement.

In this paper, we present some of the preliminary findings and strategy recommendations of the NASA Earth Science Software Reuse Working Group. First, we summarize the state of software reuse in the literature followed by an examination of current reuse practices in the earth science community. We conclude with some potential strategies for improving reuse adoption based on a discussion of issues that have to be addressed in order for a reuse process to be successful.

II. Software Reuse in the Literature

Reuse is the reapplication of various kinds of knowledge about one system to another system in order to reduce the effort of developing and maintaining that system. The reused knowledge includes such things as domain knowledge, technology expertise and development experience. This knowledge is embedded in the various development artifacts produced during the software development process, such as analysis models, design documentation and program source code. The motivation for reuse is typically based on productivity and quality improvements. Productivity is often defined as a function of cost and labor. If reuse can save cost and labor compared to developing software from scratch, then it enhances productivity. Maintenance effort is, in part, dependent on the system's defect rate. If, by reusing previously tested and debugged software, the overall defect rate can be reduced then maintenance effort can also be reduced.

Software reusability is the extent to which a software component can be used in multiple problem solutions. The reliability of reusable artifacts has a direct impact on the quality of the system that is reusing them. In general, to be reusable, artifacts must be designed and implemented accordingly. Fig. 1 summarizes several recommended properties for a reusable component [1]:

- Self-contained: embodies only a single idea or set of closely related ideas
- Additivity: able to combine components with minimal side effects
- Formal mathematical basis: allow correctness conditions to be stated and component combination to preserve key properties of components
- Confidence: the (subjective) probability that a module, program or system performs its defined purpose satisfactorily (without failure) over a specified time in another environment than it was originally constructed and/or certified for
- Understandability: its purpose is clear to the inspector
- Verifiable: easy to test
- Encapsulation: internal elements, in particular data structures, are hidden
- Simple interface: minimal number of parameters passed and parameters passed explicitly
- Flexibility: the existence of a range of choices available to the implementer
- Easily changed: easy to modify with minimal and obvious side effects
- Generality: generic functionality within a particular domain
- Programming language-independent: not unnecessarily specific about superficial language details
- Portability: can be transferred from one computer system or environment to another

Figure 1. Recommended properties of a reusable component.

I. *Reusability Methods*

Fig. 2 summarizes the two common types of code reuse:

- **Black-box reuse** is reuse without modification. It can assist consistency across products and reduce redundant development and maintenance efforts. The problem with this method is that many reusable components are ignored because they do not meet the exact needs of the target system.
- **White-box reuse** is reuse with modification. It is more popular with many implementers because the component can be tailored to fit the exact needs of the target system. However, the productivity benefits of reuse can rapidly diminish as more modifications are introduced. In addition, changes to the code can introduce errors and other unexpected side effects.

Figure 2. Common types of code reuse.

Other terms have been coined to describe variants of these two approaches: for example *gray box reuse*, where the implementer has the ability to customize only selected parts of the component, and *glass box reuse*, where the implementer can examine the contents of the component to get a better understanding of how it works but cannot make changes.

II. *Software Reuse Process*

There are generally two major aspects of software reuse to consider: building and using reusable components. The steps to take for software reuse and the issues to consider for each aspect are summarized in the following table [2]:

TABLE I. SOFTWARE REUSE ASPECTS

Building Reusable Components	***Using Reusable Components***
Identify components that can be widely reused Define components and adapting them for reuse • *Domain analysis* Classifying and storing the reusable components in a library • *Hierarchical organization scheme by application type and by function within the application* • *Object-oriented techniques such as inheritance and classes* Represent reusable components in a standard form • *Graphical format* • *Hypertext system* • *Object-oriented programming languages (encapsulation, inheritance, abstract data types, object classes)* • *Testing tools* • *Metrics tools* • *Standard checkers*	Find the reusable components • *Components need to be easy to find* • *Search methods such as keyword in context (KWIC), function index , hierarchy of function categories , keyword index, structured queries, pattern matching* Understand the reusable components • *Re-Engineering tools such as program analyzer, logic and data tracer, logic and data restructures, logic and data reverse engineering* • *Hypertext system that links reusable components to documentation* Modify the reusable components • "When" and "how" used information • Program code analyzer • Metrics tools • Other tools to validate completeness, consistency, compliance to standards and the quality of the modified components Combine and incorporate the reusable components • *CASE tools*

To facilitate the software reuse process, developers need to be able to easily locate and evaluate the available reusable artifacts. For this reason, most studies suggest that the reusable artifacts should be classified and made available through an appropriate clearinghouse (i.e., libraries, repositories) that can facilitate searching and indexing. These catalogs and repositories are an essential ingredient in transforming ad-hoc reuse, which is largely dependent on personal knowledge and word of mouth dissemination of information about the availability of reusable artifacts, to reuse as a systematic part of the software development process. Component-based architectures [6, 7] as well as technologies such as web services, semantic web, conceptual graphs and domain ontologies [8, 9, 10] can also be used to support this process and improve the classification and retrieval of reusable components.

Another key ingredient in systematizing software reuse is the adaptation of the software development process. By incorporating reuse and reusability assessments at appropriate stages in a project's software development life cycle, the project establishes formal decision making points for evaluating and validating reuse and building for reuse decisions. The reuse of a particular software component can have a significant impact on the design and implantation of the other parts of the system that need to interact with it, so it is important to identify any reusable components as early as possible in the development process.

III. THE EARTH SCIENCE DATA SYSTEMS SOFTWARE REUSE WORKING GROUP

To address the technical issues required to enable and facilitate reuse of those software assets within NASA's Earth Science Enterprise (ESE), the NASA Earth Science Software Reuse Working Group was created. The Working Group was chartered to oversee the process that will maximize the reuse potential of such components in order to (1) drive down the cost and time of system development, and reduce/eliminate unnecessary duplication of effort; (2) increase flexibility and responsiveness relative to Earth Science community needs and technological opportunities; and (3) increase effective and accountable community participation.

A. *Software Reuse Working Group Goals*

The Working Group is currently recommending and supporting activities that help increase awareness of available components, increase awareness of the value of reuse, provide needed processes and mechanisms, disseminate successful reuse strategies, and address related intellectual property and policy issues. In the process, the Working Group is considering a variety of approaches to enabling reuse to help meet differing needs and priorities across various Earth science systems.

B. *Approach*

To achieve the above goals, the Working Group is engaging in the following activities:

- Providing technical consultation, as needed, in reuse implementation projects and other efforts that directly result in the publication or use of reusable components including the registration and categorization of reusable components

- Leading outreach and education activities and sponsoring efforts that increase community awareness and understanding of reuse benefits, pro-actively foster cooperation within the community, as well as facilitate the exchange of best practices, lessons learned, tools, available components, etc.

- Providing technical consultation, as required, in support and enablement activities which include supporting infrastructure building efforts and other mechanisms needed to enable reuse (tools, metadata mining, etc)
- Contributing to policy change activities, especially those related to reducing policy barriers to reuse.

More information about the Working Group and its progress to-date can be found in [3].

IV. Software reuse survey and preliminary results

To learn about the software reuse trends and needs in the Earth Science community, the Working Group designed a survey to capture the components reused by the community in the recent past (success stories) and the components that they would like to reuse in the near future (near-term needs). Below is an overview of the elements of the survey as well as some preliminary results that will help in setting the direction of the group and identifying the strategies for enabling reuse within the community.

A. Survey

A software reuse survey was distributed to members of the Earth Science community, and consisted of the following parts

- Information about respondent (role in software development, type of organization, operating systems and programming languages)
- Recent reuse experiences (experiences using software development artifacts developed outside of project/group, types of artifacts reused, reuse percentage, barriers to reuse, reasons for considering reuse, factors influencing decisions for reusing existing assets, and types of licensing exercised)
- Recent reusability experiences (experiences developing software for reuse, types of artifacts provided for reuse and barriers for developing for reuse)
- Community needs (factors that can increase the level of reuse within the community and new approaches considered by the community)

B. Preliminary Results

Since the survey is still open, we only present here some preliminary results. Further analysis of the responses will be presented in a future paper.

1) Reuse Experiences

To-date, the survey revealed that source code and scripts as well as algorithms and techniques have been the most commonly reused types of artifacts within the last five years. When asked about the factors influencing their decision to consider reuse, most respondents chose saving time/money and ensuring reliability as their primary drivers for reuse. Most respondents also indicated that ease of adaptation/integration, availability of source code and cost of creating/acquiring alternative were the key factors for evaluating any given artifact for reuse. Perhaps surprisingly, the availability of support/maintenance, standards compliance and testing/certification were not ranked as particularly important by the respondents, whereas a recommendation from a colleague was.

Furthermore, the ranking of sources used to locate reusable artifacts confirmed some of our earlier interviews: most reuse has been reliant on identifying artifacts found through word of mouth or personal knowledge from past projects. Generic search tools (such as Google) were rated as somewhat important, whereas specialist reuse catalogs or repositories were not cited as being particularly important. The latter probably reflects a large number of disparate catalogs and repositories and the absence of a catalog specifically targeted at the Earth Science community.

Finally, in response to the question about why respondents chose not to reuse an existing artifact, a variety of barriers to reuse were identified. The wide variety of responses would seem to reflect the range of individual experiences pertaining to each individual reuse instance. However two common themes emerged: (1) available software did not exactly meet the reuser’s requirements, (2) the software was difficult to understand or poorly documented.

2) Reusability Experiences

Most of the respondents (80%) claim to have made some of their software development artifacts available for reuse outside of their immediate project. However, the additional cost of developing for reuse and concerns over support and maintenance were identified as factors that may prevent more artifacts from being made available. Amongst those respondents that had not made artifacts available for reuse, their organizations’ software release policies, concerns over intellectual property rights and, in particular, the absence of a common distribution mechanism were regarded as additional barriers to making artifacts available to others.

3) Community Needs

When asked about the factors that can help remove some of the above barriers and increase reuse within the Earth Science community, there was significant support for an increased use of open source licensing, establishing an Earth Science-focused catalog or repository for reusable artifacts and for providing education/guidance on reuse.

V. ESE assets classification model

In preparation for investigating alternatives for an ESE-focused repository or catalog, it was necessary for the Working Group to develop an ESE-specific assets classification model to be used as a basis for a future reuse enablement system for the community.

A. Assets Classification Model: Domain Analysis Approach

The literature indicates that reusable software components can be classified into two categories: horizontal and vertical. Horizontal reuse refers to reuse across a broad range of application areas such as user-interface, data structure and sorting algorithms. Vertical components, on the other hand, can be reused in a similar application within the same problem domain. Since vertical reuse is desired in the case of the Earth Science community, the Working Group determined that a domain analysis study is required. Domain analysis is defined as a process by which information used in developing software systems is identified, captured and organized with the purpose of making it reusable when creating new systems. Domain analysis deals with the development and evolution of an information infrastructure in support of reuse. Fig. 3 summarizes the generic activities in domain analysis methods [4]:

- Domain characterization and project planning (includes selection and description of domain, identification of relevant data, creation of data inventory and project planning)
- Data collection (includes recovering abstraction, reviewing the literature, eliciting knowledge from experts and developing scenarios)
- Data analysis (includes identification of entities, events, operations and relationships, modularization of the information, analysis of similarities, variations, trade-offs and combinations
- Classification (includes cluster, abstract, classification, generalization descriptions and construction of vocabulary)
- Evaluation of domain model

Figure 3. Typical domain analysis activities.

Table II captures the high level classification themes as drafted by the Working Group in preparation for the development of a formal classification model.

TABLE II. HIGH LEVEL EARTH SCIENCE CLASSIFICATION THEMES

Application Layer	***Function***	***Examples***
User Application Layer	Visualization	GUI, Output format (GIF, HDF-EOS, Polygon, TIFF etc.)
	Analysis	Image Analysis, Clustering, Pattern Recognition, Spatial Filtering, Texture Operations
Science Data Processing Layer	Discovery and Analysis	Data Product identification, Sampling, Access Control, Metadata Search Capabilities, Metadata Transport, Data Product Transports
	Processing	Data Selection, Validation, Dependencies, Ingest, Archive, dissemination, Process Execution Planning and Scheduling Activity Monitoring, Resource Scheduling and Optimization
Data Archive Layer	Management	Data Transport, Metadata Definition, Data Format
	Storage	Storage Abstraction: File Systems, Databases, HSM

Such a classification model can be the basis for a reference architecture, which has also been shown to help in enabling software reuse within a community [5]. Such a reference architecture captures the fundamental components of the domain and the relations between them. Many mature domains, such as compilers and operating systems, have well-known reference architectures that have facilitated the software reuse.

VI. STRATEGIES AND NEXT STEPS

Based on the literature research and Earth Science community-specific results presented in this paper, the Working Group will continue to find ways to support software reuse in the form of

- Building a support structure that would enable and facilitate software reuse within the Earth Science community.
- Studying domain-specific classification schemes that can assist with indexing, searching and retrieving individual ESE software artifacts efficiently.
- Investigating requirements for a reuse enablement system that goes beyond Earth Science-specific repositories or catalogs.
- Identifying high-quality reusable components and making them available for others to reuse.
- Working on a commitment from the project sponsors (in the form of funding, incentives and policy changes) and from the community (in the form of participation in the working group and the contribution/use of reusable assets).

The reader is referred to the Working Group web page [3] for updates on the activities of the Working Group and its progress in defining reuse enablement strategies and in achieving the above objectives.

REFERENCES

[1] R. Martin, G. Jackoway, and C. Ranganathan,, “Software Reuse Across Continents”, Hewlett-Packard, Position Paper, 1994.

[2] C. McClure, The Three Rs of Software Engineering: Re-Engineering, Repository, Reusability. Prentice-Hall Inc. 1992.

[3] NASA Data Systems Reuse Working Group Web Page http://softwarereuse.gsfc.nasa.gov.

[4] J.M. Armstrong, and R. J. Mitchell, “Uses and Abuses of Inheritance”, Software Engineering Journal, Vol.: 9 Iss: 1, pp. 19-26, Jan. 1994.

[5] A.E. Hassan and R.C. Holt, “A reference architecture for Web servers”, Reverse Engineering, 2000”. Proceedings. Seventh Working Conference, pp. 150--159, Nov. 2000.

[6] H. Yao and L. Etzkorn, “Towards a semantic-based approach for software reusable component classification and retrieval”, Proceedings of the 42nd annual Southeast regional conference SESSION: Software engineering #1, pp. 110 – 115, 2004

[7] Y. Kim, E. A. Stohr, "Software Reuse: Survey and Research Directions," Journal of Management Information Systems, Spring 1998, Vol. 14, No. 4, pp. 113--147.. 1998.

[8] Semantic Web (SW) http://www.w3.org/2001/sw/, accessed Oct. 13, 2003

[9] Web Service Description Language (WSDL) http://www.w3.org/TR/wsdl, accessed Oct. 12, 2003.

[10] eXtensible Markup Language (XML) http://www.w3.org/XML, accessed Oct. 12, 2003.

[11] NASA JPL SWEET Project http://sweet.jpl.nasa.gov/index.htm